**In the Claims:**

Please amend Claims 12, 17-19, 22, 27, and 31; cancel Claims 21 and 30; and add new Claims 32-33, all as shown below. Applicant respectfully reserves the right to prosecute any originally presented or canceled claims in a continuing or future application.

1-11 (Canceled).

12. (Currently Amended) A system for interleaving resource enlistment synchronization, comprising:

an application server with a plurality of threads, running on one or more processors;

a transaction manager, wherein the transaction manager maintains an enlistment data structure to manage resource object enlistment in ~~one or more~~ a plurality of transactions, wherein each transaction is associated with one or more different said threads, and wherein the enlistment data structure maintains a mapping of resource and transaction identification information;

one or more resource objects, wherein each resource object is wrapped with a wrapper object, wherein the transaction manager uses the wrapper object to synchronize concurrent enlistment requests from different said threads associated with different transactions;

wherein, after the transaction manager receives a request from one thread of the plurality of threads to enlist one of the one or more resource objects in one transaction, the transaction manager

first checks to see if there is a lock being held on the resource object by another thread in another transaction;

if not, grants a lock to an accessor associated with the thread and holds the lock until an owner of the thread delists the resource object.

13. (Previously Presented) The system of Claim 12, wherein:

the transaction manager maintains a collection of wrapper objects that wraps the one or more resource objects.

14. (Previously Presented) The system of Claim 13, wherein:

the collection of wrapper objects is periodically processed to remove objects that are unused or no longer active.

15. (Previously Presented) The system of Claim 12, wherein:

each of the one or more resource objects resides in a server node.

16. (Previously Presented) The system of Claim 12, wherein:

the transaction manager signals to one or more waiting threads once a lock is free.

17. (Currently Amended) The system of Claim 12, wherein:

the transaction manager uses a priority method to determine which thread will be granted a lock.

18. (Currently Amended) The system of Claim 12, wherein:

after the thread obtains a lock, the thread uses the wrapper object to initiate work on the resource object.

19. (Currently Amended) The system of Claim 12, wherein:

the wrapper object receives a delist call from the transaction manager and sends an end call to the resource object to end work performed by the resource object associated with the thread and release the lock on the resource object.

20. (Previously Presented) The system of Claim 12, wherein:

once the transaction manager enlists the resource object and obtains a lock to the resource object, any attempted enlist from a second thread is blocked.

21. (Canceled).

22. (Currently Amended) A method for interleaving resource enlistment synchronization, comprising:

providing an application server with a plurality of threads, running on one or more processors;

maintaining an enlistment data structure, at a transaction manager, to manage resource object enlistment in ~~one or more~~ a plurality of transactions, wherein each transaction is

associated with one or more different said threads, and wherein the enlistment data structure maintains a mapping of resource and transaction identification information;

wrapping each of one or more resource objects with a wrapper object, wherein the transaction manager uses the wrapper object to synchronize concurrent enlistment requests from different said threads associated with different transactions;

receiving a request from one thread of the plurality of threads to enlist one of the one or more resource objects in one transaction at the transaction manager;

first checking, via the transaction manager, to see if there is a lock being held on the resource object by another thread in another transaction;

if not, granting, via the transaction manager, a lock to an accessor associated with the thread and holding the lock until an owner of the thread delists the resource object.

23. (Previously Presented) The method of Claim 22, further comprising:

maintaining a collection of wrapper objects that wraps the one or more resource objects.

24. (Previously Presented) The method of Claim 23, wherein:

the collection of wrapper objects is periodically processed to remove objects that are unused or no longer active.

25. (Previously Presented) The method of Claim 22, further comprising:

signaling to one or more waiting threads once a lock is free.

26. (Previously Presented) The method of Claim 22, further comprising:

using a priority method to determine which thread will be granted a lock.

27. (Currently Amended) The method of Claim 22, wherein:

after the thread obtains a lock, the thread uses the wrapper object to initiate work on the resource object.

28. (Previously Presented) The method of Claim 22, wherein:

the wrapper object receives a delist call from the transaction manager and send an end call to the resource object to end work performed by the resource object associated with the thread and release the lock on the resource object.

29. (Previously Presented) The method of Claim 22, wherein:

once the transaction manager enlists the resource object and obtains a lock to the resource object, any attempted enlist from a second thread is blocked.

30. (Canceled).

31. (Currently Amended) A computer-readable storage medium, storing instructions for interleaving resource enlistment synchronization, the instructions comprising the steps of:

providing an application server with a plurality of threads, running on one or more processors;

maintaining an enlistment data structure, at a transaction manager, to manage resource object enlistment in ~~one or more~~ a plurality of transactions, wherein each transaction is associated with one or more different said threads, and wherein the enlistment data structure maintains a mapping of resource and transaction identification information;

wrapping each of one or more resource objects with a wrapper object, wherein the transaction manager uses the wrapper object to synchronize concurrent enlistment requests from different said threads associated with different transactions;

receiving a request from one thread of the plurality of threads to enlist one of the one or more resource objects in one transaction at the transaction manager;

first checking, via the transaction manager, to see if there is a lock being held on the resource object by another thread in another transaction;

if not, granting, via the transaction manager, a lock to an accessor associated with the thread and holding the lock until an owner of the thread delists the resource object.

32. (New) The system of Claim 12, wherein:

the wrapper object is periodically garbage collected to clear state and unused locks.

33. (New) The method of Claim 22, further comprising:

determining, via the transaction manager, whether an application associated with the thread is a specific type of application;

granting, via the transaction manager, the thread a lock only when the application is determined to be the specific type of application.